Every graph is easy or hard: dichotomy theorems for graph problems

Dániel Marx[1]

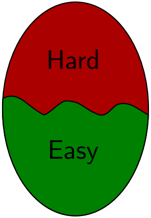
[1]Institute for Computer Science and Control,
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

Dagstuhl Seminar 14451
Schloss Dagstuhl, Germany
November 7, 2014

Dichotomy theorems

What is better than proving one nice result?
Proving an infinite set of nice results.

We survey results where we can precisely tell which graphs make the problem easy and which graphs make the problem hard.



Focus will be on

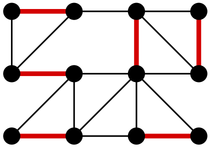
- how to formulate questions that lead to such results and
- what results of this type are known,

but less on how to prove such results.

Factor problems

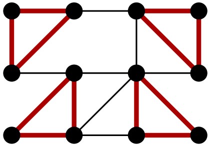
PERFECT MATCHING
Input: graph G.
Task: find $|V(G)|/2$ vertex-disjoint edges.

Polynomial-time solvable [Edmonds 1961].



TRIANGLE FACTOR
Input: graph G.
Task: find $|V(G)|/3$ vertex-disjoint triangles.

NP-complete [Karp 1975]



Factor problems

H-FACTOR
Input: graph G.
Task: find $|V(G)|/|V(H)|$ vertex-disjoint copies of H in G.

Polynomial-time solvable for $H = K_2$ and NP-hard for $H = K_3$.

Which graphs H make H-FACTOR easy and which graphs make it hard?

Factor problems

> H-FACTOR
> **Input:** graph G.
> **Task:** find $|V(G)|/|V(H)|$ vertex-disjoint copies of H in G.

Polynomial-time solvable for $H = K_2$ and NP-hard for $H = K_3$.

Which graphs H make H-FACTOR easy and which graphs make it hard?

Theorem [Kirkpatrick and Hell 1978]

H-FACTOR is NP-hard for every connected graph H with at least 3 vertices.

Factor problems

Instead of publishing

Kirkpatrick and Hell: NP-completeness of packing cycles. 1978.
Kirkpatrick and Hell: NP-completeness of packing trees. 1979.
Kirkpatrick and Hell: NP-completeness of packing stars. 1980.
Kirkpatrick and Hell: NP-completeness of packing wheels. 1981.
Kirkpatrick and Hell: NP-completeness of packing Petersen graphs. 1982.
Kirkpatrick and Hell: NP-completeness of packing Starfish graphs. 1983.
Kirkpatrick and Hell: NP-completeness of packing Jaws. 1984.

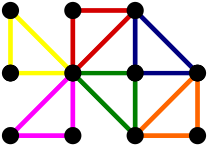
$\vdots$

they only published

Kirkpatrick and Hell: On the Completeness of a Generalized Matching Problem. 1978

Edge-disjoint version

H-DECOMPOSITION
Input: graph G.
Task: find $|E(G)|/|E(H)|$ **edge-disjoint** copies of H in G.



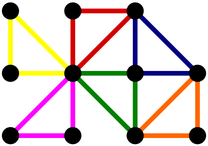
- Trivial for $H = K_2$.
- Can be solved by matching for P_3 (path on 3 vertices).

Theorem [Holyer 1981]

H-DECOMPOSITION is NP-complete if H is the clique K_r or the cycle C_r for some $r \geq 3$.

Edge-disjoint version

H-DECOMPOSITION
Input: graph G.
Task: find $|E(G)|/|E(H)|$ **edge-disjoint** copies of H in G.



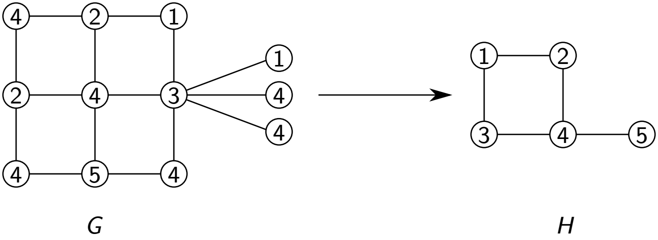
- Trivial for $H = K_2$.
- Can be solved by matching for P_3 (path on 3 vertices).

Theorem (Holyer's Conjecture) [Dor and Tarsi 1992]

H-DECOMPOSITION is NP-complete for every connected graph H with at least 3 edges.

H-coloring

A **homomorphism** from G to H is a mapping $f\colon V(G) \to V(H)$ such that if ab is an edge of G, then $f(a)f(b)$ is an edge of H.

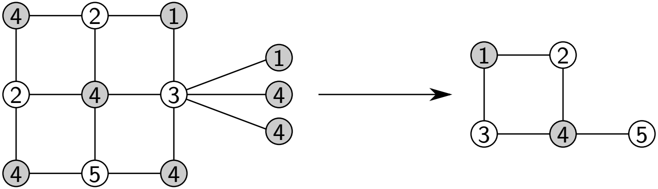


> H-COLORING
> **Input:** graph G.
> **Task:** Find a homomorphism from G to H.

- If $H = K_r$, then equivalent to r-COLORING.
- If H is bipartite, then the problem is equivalent to G being bipartite.

H-coloring

A **homomorphism** from G to H is a mapping $f: V(G) \to V(H)$ such that if ab is an edge of G, then $f(a)f(b)$ is an edge of H.



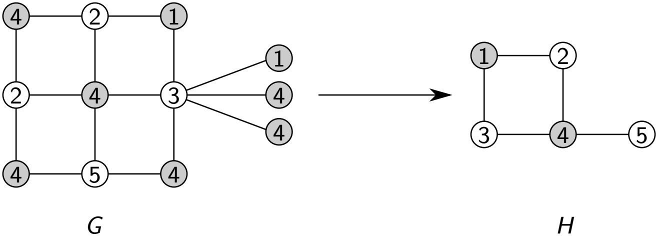
G H

H-COLORING
Input: graph G.
Task: Find a homomorphism from G to H.

- If $H = K_r$, then equivalent to r-COLORING.
- If H is bipartite, then the problem is equivalent to G being bipartite.

H-coloring

A **homomorphism** from G to H is a mapping $f\colon V(G) \to V(H)$ such that if ab is an edge of G, then $f(a)f(b)$ is an edge of H.



> H-COLORING
> **Input:** graph G.
> **Task:** Find a homomorphism from G to H.

Theorem [Hell and Nešetřil 1990]

For every simple graph H, H-COLORING is polynomial-time solvable if H is bipartite and NP-complete if H is not bipartite.

Dichotomy theorems

Dichotomy theorem: classifying every member of a family of problems as easy or hard.

Why are such theorems surprising?

1. The characterization of easy/hard is a simple combinatorial property.

   So far, we have seen:

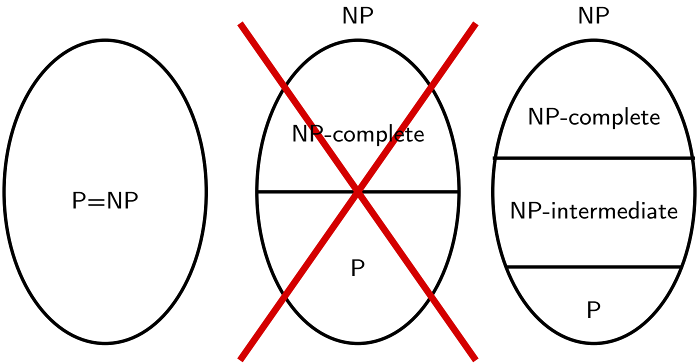
   - at least 3 vertices,
   - nonbipartite.

Dichotomy theorems

2. Every problem is either in P or NP-complete, there are no NP-intermediate problems in the family.

Theorem [Ladner 1973]

If $P \neq NP$, then there is language $L \in NP \setminus P$ that is not NP-complete.



Dichotomy theorems

- Dichotomy theorems give goods research programs: easy to formulate, but can be hard to complete.
- The search for dichotomy theorems may uncover algorithmic results that no one has thought of.
- Proving dichotomy theorems may require good command of both algorithmic and hardness proof techniques.

Dichotomy theorems

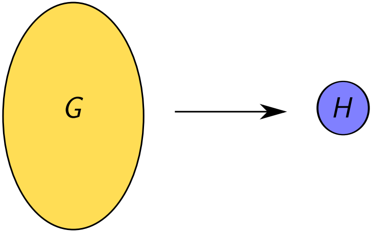
- Dichotomy theorems give goods research programs: easy to formulate, but can be hard to complete.
- The search for dichotomy theorems may uncover algorithmic results that no one has thought of.
- Proving dichotomy theorems may require good command of both algorithmic and hardness proof techniques.

So far:
Each problem in the family was defined by fixing a graph H.

Next:
Each problem is defined by fixing a **class** of graph $\mathcal{H}$.

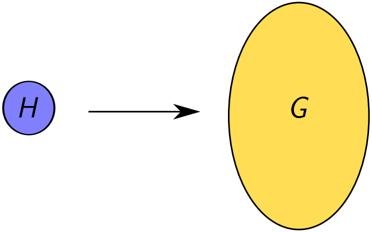
Homomorphisms seen from the other side

Recall: H-COLORING (finding a homomorphism to H) is polynomial-time solvable if H is bipartite and NP-complete otherwise.



Homomorphisms seen from the other side

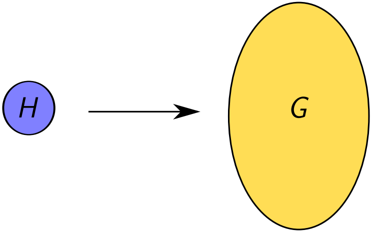
Recall: H-COLORING (finding a homomorphism to H) is polynomial-time solvable if H is bipartite and NP-complete otherwise.



What about finding a homomorphism *from* H?

Homomorphisms seen from the other side

Recall: H-COLORING (finding a homomorphism to H) is polynomial-time solvable if H is bipartite and NP-complete otherwise.



What about finding a homomorphism *from* H?

Theorem (trivial)

For every fixed H, the problem $\textsc{Hom}(H, -)$ (find a homomorphism from H to the given graph G) is polynomial-time solvable.

… because we can try all $|V(G)|^{|V(H)|}$ possible mappings $f\colon V(H) \to V(G)$.

Homomorphisms seen from the other side

Better question:

$\textsc{Hom}(\mathcal{H}, -)$ **Input:** a graph $H \in \mathcal{H}$ and an arbitrary graph G. **Task:** decide if there is a homomorphism from H to G.

Goal: characterize the classes $\mathcal{H}$ for which $\textsc{Hom}(\mathcal{H}, -)$ is polynomial-time solvable.

For example, if $\mathcal{H}$ contains only bipartite graphs, then $\textsc{Hom}(\mathcal{H}, -)$ is polynomial-time solvable.

Homomorphisms seen from the other side

Better question:

$\textsc{Hom}(\mathcal{H}, -)$ **Input:** a graph $H \in \mathcal{H}$ and an arbitrary graph G. **Task:** decide if there is a homomorphism from H to G.

Goal: characterize the classes $\mathcal{H}$ for which $\textsc{Hom}(\mathcal{H}, -)$ is polynomial-time solvable.

For example, if $\mathcal{H}$ contains only bipartite graphs, then $\textsc{Hom}(\mathcal{H}, -)$ is polynomial-time solvable.

We have reasons to believe that there is no P vs. NP-complete dichotomy for $\textsc{Hom}(\mathcal{H}, -)$. Instead of NP-completeness, we will use parameterized complexity for giving negative evidence.

Counting homomorphisms

$\#\textsc{Hom}(\mathcal{H}, -)$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: count the number of homomorphisms from $H \to G$.

We parameterize by $k = |V(H)|$, i.e., our goal is an $f(|V(H)|) \cdot n^{O(1)}$ time algorithm.

Counting homomorphisms

$\#\textsc{Hom}(\mathcal{H}, -)$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: count the number of homomorphisms from $H \to G$.

We parameterize by $k = |V(H)|$, i.e., our goal is an $f(|V(H)|) \cdot n^{O(1)}$ time algorithm.

Theorem [Dalmau and Jonsson 2004]

Assuming FPT $\neq$ W[1], for every recursively enumerable class $\mathcal{H}$ of graphs, the following are equivalent:

1. $\#\textsc{Hom}(\mathcal{H}, -)$ is polynomial-time solvable.
2. $\#\textsc{Hom}(\mathcal{H}, -)$ is FPT parameterized by $|V(H)|$.
3. $\mathcal{H}$ has bounded treewidth.

Counting homomorphisms

$\#\textsc{Hom}(\mathcal{H}, -)$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: count the number of homomorphisms from $H \to G$.

We parameterize by $k = |V(H)|$, i.e., our goal is an $f(|V(H)|) \cdot n^{O(1)}$ time algorithm.

Theorem [Dalmau and Jonsson 2004]

Assuming FPT $\neq$ W[1], for every recursively enumerable class $\mathcal{H}$ of graphs, the following are equivalent:

1. $\#\textsc{Hom}(\mathcal{H}, -)$ is polynomial-time solvable.
2. $\#\textsc{Hom}(\mathcal{H}, -)$ is FPT parameterized by $|V(H)|$.
3. $\mathcal{H}$ has bounded treewidth.

Excluded Grid Theorem [Robertson and Seymour]

There is a function f such that every graph with treewidth $f(k)$ contains a $k \times k$ grid minor.

Counting homomorphisms

$\#\textsc{Hom}(\mathcal{H}, -)$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: count the number of homomorphisms from $H \to G$.

We parameterize by $k = |V(H)|$, i.e., our goal is an $f(|V(H)|) \cdot n^{O(1)}$ time algorithm.

Theorem [Dalmau and Jonsson 2004]

Assuming FPT $\neq$ W[1], for every recursively enumerable class $\mathcal{H}$ of graphs, the following are equivalent:

1. $\#\textsc{Hom}(\mathcal{H}, -)$ is polynomial-time solvable.
2. $\#\textsc{Hom}(\mathcal{H}, -)$ is FPT parameterized by $|V(H)|$.
3. $\mathcal{H}$ has bounded treewidth.

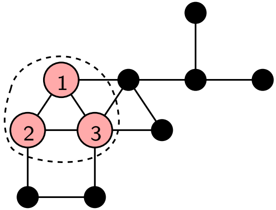
Steps of the proof:

- Show that the problem is polynomial-time solvable for bounded treewidth.
- Show that the problem is W[1]-hard if $\mathcal{H}$ is the class of grids.

Decision version

> $\textsc{Hom}(\mathcal{H}, -)$
> **Input:** a graph $H \in \mathcal{H}$ and an arbitrary graph G.
> **Task:** decide if there is a homomorphism from H to G.

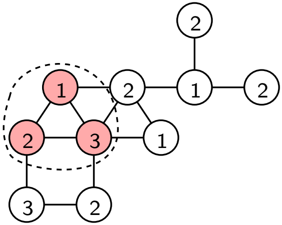
Core of H**:** smallest subgraph H^* of H such that there is a homomorphism $H \to H^*$ (known to be unique up to isomorphism).



Decision version

$\textsc{Hom}(\mathcal{H}, -)$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: decide if there is a homomorphism from H to G.

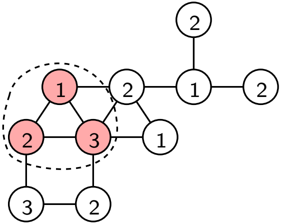
Core of H: smallest subgraph H^* of H such that there is a homomorphism $H \to H^*$ (known to be unique up to isomorphism).



Decision version

$\textsc{Hom}(\mathcal{H}, -)$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: decide if there is a homomorphism from H to G.

Core of H: smallest subgraph H^* of H such that there is a homomorphism $H \to H^*$ (known to be unique up to isomorphism).



Observation

If H^* is the core of H, then there is a homomorphism $H^* \to G$ if and only if there is a homomorphism $H \to G$.

Decision version

$\textsc{Hom}(\mathcal{H}, -)$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: decide if there is a homomorphism from H to G.

Core of H: smallest subgraph H^* of H such that there is a homomorphism $H \to H^*$ (known to be unique up to isomorphism).

Theorem [Grohe 2003]

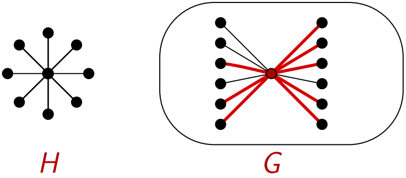
Assuming $\mathrm{FPT} \neq \mathrm{W}[1]$, for every recursively enumerable class $\mathcal{H}$ of graphs, the following are equivalent:

1. $\textsc{Hom}(\mathcal{H}, -)$ is polynomial-time solvable.
2. $\textsc{Hom}(\mathcal{H}, -)$ is FPT parameterized by $|V(H)|$.
3. there is a constant $c \geq 1$ such that the core of every graph in $\mathcal{H}$ has treewidth at most c.

Counting subgraphs

$\#\mathrm{SUB}(\mathcal{H})$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: calculate the number of copies of H in G.

If $\mathcal{H}$ is the class of all stars, then $\#\mathrm{SUB}(\mathcal{H})$ is easy: for each placement of the center of the star, calculate the number of possible different assignments of the leaves.

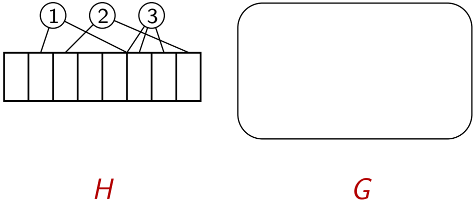


Counting subgraphs

$\#\text{SUB}(\mathcal{H})$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: calculate the number of copies of H in G.

Theorem

If every graph in $\mathcal{H}$ has vertex cover number at most c, then $\#\text{SUB}(\mathcal{H})$ is polynomial-time solvable.



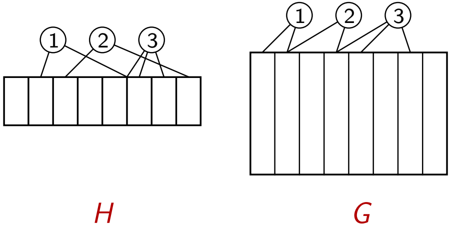
Running time is $n^{2^{O(c)}}$, better algorithms known [Vassilevska Williams and Williams], [Kowaluk, Lingas, and Lundell].

Counting subgraphs

$\#\textsc{Sub}(\mathcal{H})$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: calculate the number of copies of H in G.

Theorem

If every graph in $\mathcal{H}$ has vertex cover number at most c, then $\#\textsc{Sub}(\mathcal{H})$ is polynomial-time solvable.



Running time is $n^{2^{O(c)}}$, better algorithms known [Vassilevska Williams and Williams], [Kowaluk, Lingas, and Lundell].

Counting subgraphs

Who are the bad guys now?

Theorem [Flum and Grohe 2002]

If $\mathcal{H}$ is the set of all paths, then $\#\text{Sub}(\mathcal{H})$ is #W[1]-hard.

Theorem [Curticapean 2013]

If $\mathcal{H}$ is the set of all matchings, then $\#\text{Sub}(\mathcal{H})$ is #W[1]-hard.

Counting subgraphs

Who are the bad guys now?

Theorem [Flum and Grohe 2002]

If $\mathcal{H}$ is the set of all paths, then $\#\text{Sub}(\mathcal{H})$ is #W[1]-hard.

Theorem [Curticapean 2013]

If $\mathcal{H}$ is the set of all matchings, then $\#\text{Sub}(\mathcal{H})$ is #W[1]-hard.

Dichotomy theorem:

Theorem [Curticapean and M. 2014]

Let $\mathcal{H}$ be a recursively enumerable class of graphs. If $\mathcal{H}$ has unbounded vertex cover number, then $\#\text{Sub}(\mathcal{H})$ is #W[1]-hard.

($\nu(G) \le \tau(G) \le 2\nu(G)$, hence "unbounded vertex cover number" and "unbounded matching number" are the same.)

Counting subgraphs

Who are the bad guys now?

Theorem [Flum and Grohe 2002]

If $\mathcal{H}$ is the set of all paths, then $\#\textsc{Sub}(\mathcal{H})$ is #W[1]-hard.

Theorem [Curticapean 2013]

If $\mathcal{H}$ is the set of all matchings, then $\#\textsc{Sub}(\mathcal{H})$ is #W[1]-hard.

Dichotomy theorem:

Theorem [Curticapean and M. 2014]

Let $\mathcal{H}$ be a recursively enumerable class of graphs. If $\mathcal{H}$ has unbounded vertex cover number, then $\#\textsc{Sub}(\mathcal{H})$ is #W[1]-hard.

($\nu(G) \le \tau(G) \le 2\nu(G)$, hence "unbounded vertex cover number" and "unbounded matching number" are the same.)

There is a simple proof if $\mathcal{H}$ is hereditary, but the general case is more difficult.

Counting subgraphs

Observation

At least one of the following holds for every hereditary class $\mathcal{H}$ with unbounded vertex cover number:

- $\mathcal{H}$ contains every matching.
- $\mathcal{H}$ contains every clique.
- $\mathcal{H}$ contains every biclique.

Counting subgraphs

Observation

At least one of the following holds for every hereditary class $\mathcal{H}$ with unbounded vertex cover number:

- $\mathcal{H}$ contains every matching. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every clique. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every biclique. $\Rightarrow$ #W[1]-hard

Counting subgraphs

Observation

At least one of the following holds for every hereditary class $\mathcal{H}$ with unbounded vertex cover number:

- $\mathcal{H}$ contains every matching. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every clique. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every biclique. $\Rightarrow$ #W[1]-hard

Ramsey's Theorem: There is a monochromatic r-clique in every c-coloring of the edges of a clique with at least c^{cr} vertices.

Counting subgraphs

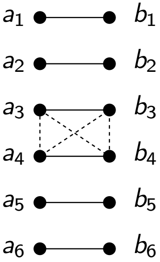
Observation

At least one of the following holds for every hereditary class $\mathcal{H}$ with unbounded vertex cover number:

- $\mathcal{H}$ contains every matching. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every clique. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every biclique. $\Rightarrow$ #W[1]-hard

Ramsey's Theorem: There is a monochromatic r-clique in every c-coloring of the edges of a clique with at least c^{cr} vertices.

- For every $i < j$, there are 2^4 possibilities for the 4 edges between $\{a_i, b_i\}$ and $\{a_j, b_j\}$.
- If there is a large matching, then there is a large matching that is homogeneous with respect to these 16 possibilities.



Counting subgraphs

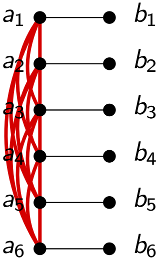
Observation

At least one of the following holds for every hereditary class $\mathcal{H}$ with unbounded vertex cover number:

- $\mathcal{H}$ contains every matching. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every clique. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every biclique. $\Rightarrow$ #W[1]-hard

Ramsey's Theorem: There is a monochromatic r-clique in every c-coloring of the edges of a clique with at least c^{cr} vertices.

- For every $i < j$, there are 2^4 possibilities for the 4 edges between $\{a_i, b_i\}$ and $\{a_j, b_j\}$.
- If there is a large matching, then there is a large matching that is homogeneous with respect to these 16 possibilities.
- In each of the 16 cases, we find a matching, clique, or biclique as induced subgraph.



Counting subgraphs

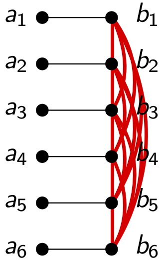
Observation

At least one of the following holds for every hereditary class $\mathcal{H}$ with unbounded vertex cover number:

- $\mathcal{H}$ contains every matching. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every clique. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every biclique. $\Rightarrow$ #W[1]-hard

Ramsey's Theorem: There is a monochromatic r-clique in every c-coloring of the edges of a clique with at least c^{cr} vertices.

- For every $i < j$, there are 2^4 possibilities for the 4 edges between $\{a_i, b_i\}$ and $\{a_j, b_j\}$.
- If there is a large matching, then there is a large matching that is homogeneous with respect to these 16 possibilities.
- In each of the 16 cases, we find a matching, clique, or biclique as induced subgraph.



Counting subgraphs

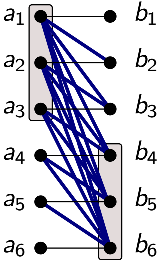
Observation

At least one of the following holds for every hereditary class $\mathcal{H}$ with unbounded vertex cover number:

- $\mathcal{H}$ contains every matching. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every clique. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every biclique. $\Rightarrow$ #W[1]-hard

Ramsey's Theorem: There is a monochromatic r-clique in every c-coloring of the edges of a clique with at least c^{cr} vertices.

- For every $i < j$, there are 2^4 possibilities for the 4 edges between $\{a_i, b_i\}$ and $\{a_j, b_j\}$.
- If there is a large matching, then there is a large matching that is homogeneous with respect to these 16 possibilities.
- In each of the 16 cases, we find a matching, clique, or biclique as induced subgraph.



Counting subgraphs

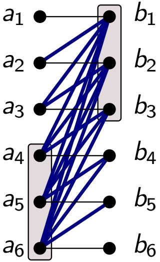
Observation

At least one of the following holds for every hereditary class $\mathcal{H}$ with unbounded vertex cover number:

- $\mathcal{H}$ contains every matching. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every clique. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every biclique. $\Rightarrow$ #W[1]-hard

Ramsey's Theorem: There is a monochromatic r-clique in every c-coloring of the edges of a clique with at least c^{cr} vertices.

- For every $i < j$, there are 2^4 possibilities for the 4 edges between $\{a_i, b_i\}$ and $\{a_j, b_j\}$.
- If there is a large matching, then there is a large matching that is homogeneous with respect to these 16 possibilities.
- In each of the 16 cases, we find a matching, clique, or biclique as induced subgraph.



Counting subgraphs

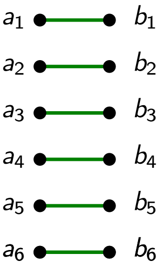
Observation

At least one of the following holds for every hereditary class $\mathcal{H}$ with unbounded vertex cover number:

- $\mathcal{H}$ contains every matching. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every clique. $\Rightarrow$ #W[1]-hard
- $\mathcal{H}$ contains every biclique. $\Rightarrow$ #W[1]-hard

Ramsey's Theorem: There is a monochromatic r-clique in every c-coloring of the edges of a clique with at least c^{cr} vertices.

- For every $i < j$, there are 2^4 possibilities for the 4 edges between $\{a_i, b_i\}$ and $\{a_j, b_j\}$.
- If there is a large matching, then there is a large matching that is homogeneous with respect to these 16 possibilities.
- In each of the 16 cases, we find a matching, clique, or biclique as induced subgraph.



H-packing

H-PACKING
Input: an arbitrary graph G and an integer k.
Task: decide if there are k vertex-disjoint copies of H in G.

Question: For which fixed graphs H the problem H-PACKING has a polynomial kernel?

H-packing

H-PACKING
Input: an arbitrary graph G and an integer k.
Task: decide if there are k vertex-disjoint copies of H in G.

Question: For which fixed graphs H the problem H-PACKING has a polynomial kernel?

- For every fixed H, there is a kernel of size $O(k^{|V(H)|})$.
- Interpret the problem as packing of $|V(H)|$-sets, then kernelization using the Sunflower Lemma.

H-packing

> H-PACKING
> **Input:** an arbitrary graph G and an integer k.
> **Task:** decide if there are k vertex-disjoint copies of H in G.

Question: For which fixed graphs H the problem H-PACKING has a polynomial kernel?

- For every fixed H, there is a kernel of size $O(k^{|V(H)|})$.
- Interpret the problem as packing of $|V(H)|$-sets, then kernelization using the Sunflower Lemma.

Better question: H is part of the input, but restricted to a class $\mathcal{H}$.

H-packing

$\mathcal{H}$-PACKING
Input: a graph $H \in \mathcal{H}$, an arbitrary graph G, and an integer k.
Task: decide if there are k vertex-disjoint copies of H in G.

Natural parameter: $k \cdot |V(H)|$, the size of the output.

Question: Which classes $\mathcal{H}$ admit a polynomial kernel?

H-packing

$\mathcal{H}$-PACKING
Input: a graph $H \in \mathcal{H}$, an arbitrary graph G, and an integer k.
Task: decide if there are k vertex-disjoint copies of H in G.

Natural parameter: $k \cdot |V(H)|$, the size of the output.

Question: Which classes $\mathcal{H}$ admit a polynomial kernel?

- If every component of every $H \in \mathcal{H}$ has size at most a, then there is a polynomial kernel.
- For every fixed b, packing $K_{b,t}$'s admits a polynomial kernel.
- If every component of every $H \in \mathcal{H}$ is a bipartite graph with at most b vertices on the smaller side, then there is a polynomial kernel.

H-packing

$\mathcal{H}$-PACKING
Input: a graph $H \in \mathcal{H}$, an arbitrary graph G, and an integer k.
Task: decide if there are k vertex-disjoint copies of H in G.

Natural parameter: $k \cdot |V(H)|$, the size of the output.

$\mathcal{H}$ is **small/thin** if every component of every $H \in \mathcal{H}$ is either of size $\leq a$ or a bipartite graph with $\leq b$ vertices on the smaller side.

Theorem [Jansen and M. 2015]

Let $\mathcal{H}$ be a hereditary graph class.

- If $\mathcal{H}$ is small/thin, then $\mathcal{H}$-PACKING admits a polynomial kernel.
- Otherwise, $\mathcal{H}$-PACKING admits no polynomial kernel, unless NP $\subseteq$ coNP/poly.

H-packing

> $\mathcal{H}$-Packing
> **Input:** a graph $H \in \mathcal{H}$, an arbitrary graph G, and an integer k.
> **Task:** decide if there are k vertex-disjoint copies of H in G.

Natural parameter: $k \cdot |V(H)|$, the size of the output.

$\mathcal{H}$ is **small/thin** if every component of every $H \in \mathcal{H}$ is either of size $\leq a$ or a bipartite graph with $\leq b$ vertices on the smaller side.

Theorem [Jansen and M. 2015]

Let $\mathcal{H}$ be a hereditary graph class.

- If $\mathcal{H}$ is small/thin, then $\mathcal{H}$-Packing admits a polynomial kernel.
- Otherwise, $\mathcal{H}$-Packing admits no polynomial kernel, unless NP $\subseteq$ coNP/poly and the problem is WK[1]-hard or Long Path-hard.

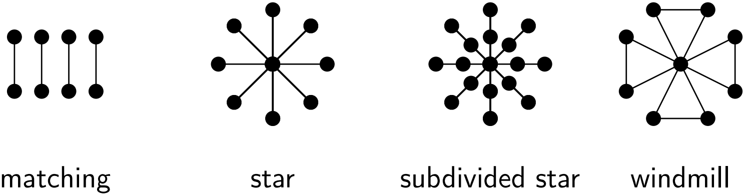
Conclusion: Turing kernels do not give us more power for any of the $\mathcal{H}$-Packing problems.

Finding subgraphs

$\textsc{Sub}(\mathcal{H})$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: decide if H is a subgraph of G.

Some classes for which $\textsc{Sub}(\mathcal{H})$ is polynomial-time solvable:

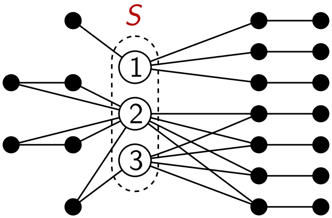
- $\mathcal{H}$ is the class of all matchings
- $\mathcal{H}$ is the class of all stars
- $\mathcal{H}$ is the class of all stars, each edge subdivided once
- $\mathcal{H}$ is the class of all windmills



Finding subgraphs

Definition

Class $\mathcal{H}$ is **matching splittable** if there is a constant c such that every $H \in \mathcal{H}$ has a set S of at most c vertices such that every component of $H - S$ has size at most 2.



Theorem [Jansen and M. 2014]

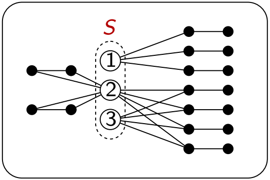
Let $\mathcal{H}$ be a hereditary class of graphs. If $\mathcal{H}$ is matching splittable, then $\textsc{Sub}(\mathcal{H})$ is randomized polynomial-time solvable and NP-hard otherwise.

Finding subgraphs (algorithm)

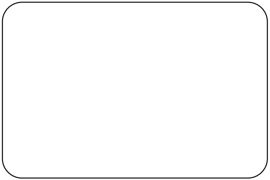
Theorem [Jansen and M. 2014]

If hereditary class $\mathcal{H}$ is matching splittable, then $\textsc{Sub}(\mathcal{H})$ is randomized polynomial-time solvable.

H



G

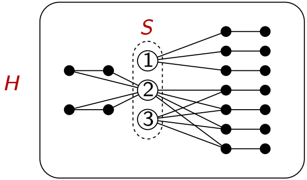


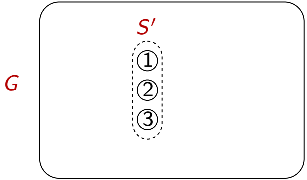
Finding subgraphs (algorithm)

Theorem [Jansen and M. 2014]

If hereditary class $\mathcal{H}$ is matching splittable, then $\textsc{Sub}(\mathcal{H})$ is randomized polynomial-time solvable.

- Guess the image S' of S in G.





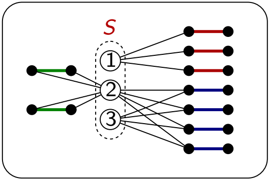
Finding subgraphs (algorithm)

Theorem [Jansen and M. 2014]

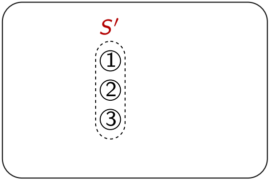
If hereditary class $\mathcal{H}$ is matching splittable, then $\textsc{Sub}(\mathcal{H})$ is randomized polynomial-time solvable.

- Guess the image S' of S in G.
- Classify the edges of $H - S$ according to their neighborhoods in S (at most 2^{2c} colors).

H



G



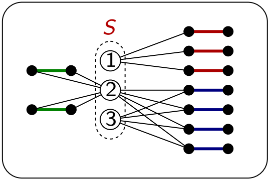
Finding subgraphs (algorithm)

Theorem [Jansen and M. 2014]

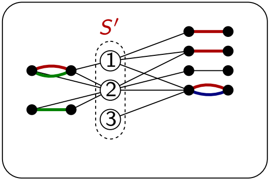
If hereditary class $\mathcal{H}$ is matching splittable, then $\textsc{Sub}(\mathcal{H})$ is randomized polynomial-time solvable.

- Guess the image S' of S in G.
- Classify the edges of $H - S$ according to their neighborhoods in S (at most 2^{2c} colors).
- Classify the edges of $G - S'$ according to which edge of $H - S$ can be mapped into it (use parallel edges if needed).
- Task is to find a matching in $G - S'$ with a certain number of edges of each color.

H



G



Finding subgraphs (algorithm)

Theorem [Mulmuley, Vazirani, Vazirani 1987]

There is a randomized polynomial-time algorithm that, given a graph G with red and blue edges and integer k, decides if there is a perfect matching with exactly k red edges.

More generally:

Theorem

Given a graph G with edges colored with c colors and c integers k_1, $\dots$, k_c, we can decide in randomized time $n^{O(c)}$ if there is a matching with exactly k_i edges of color i.

This is precisely what we need to complete the algorithm for $\textsc{Sub}(\mathcal{H})$ for matching splittable $\mathcal{H}$.

Finding subgraphs (hardness proof)

Lemma

Let $\mathcal{H}$ be a hereditary class of graphs that is not matching splittable. Then at least one of the following is true.

- $\mathcal{H}$ contains every clique.
- $\mathcal{H}$ contains every biclique.
- For every $n \geq 1$, $\mathcal{H}$ contains $n \cdot K_3$.
- For every $n \geq 1$, $\mathcal{H}$ contains $n \cdot P_3$ (where P_3 is the path on 3 vertices).

In each case, $\textsc{Sub}(\mathcal{H})$ is NP-hard (recall that P_3-$\textsc{factor}$ and K_3-$\textsc{factor}$ are NP-hard).

Finding subgraphs (hardness proof)

Recall: Class $\mathcal{H}$ is **matching splittable** if there is a constant c such that every $H \in \mathcal{H}$ has a set S of at most c vertices such that every component of $H - S$ has size at most 2.

Equivalently: in every $H \in \mathcal{H}$, we can cover every 3-vertex connected set (i.e., every K_3 and P_3) by c vertices.

Observation: either

- there are r vertex disjoint K_3, or
- there are r vertex disjoint P_3, or
- we can cover every K_3 and every P_3 by $6r$ vertices.

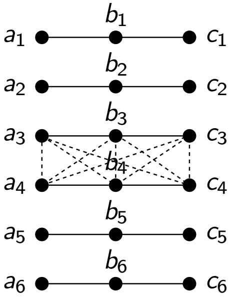
Finding subgraphs (hardness proof)

Lemma

Let $\mathcal{H}$ be a hereditary class of graphs that is not matching splittable. Then at least one of the following is true.

- $\mathcal{H}$ contains every clique.
- $\mathcal{H}$ contains every biclique.
- For every $n \geq 1$, $\mathcal{H}$ contains $n \cdot K_3$.
- For every $n \geq 1$, $\mathcal{H}$ contains $n \cdot P_3$.

- Consider many vertex-disjoint P_3's.
- For every $i < j$, there are 2^9 possibilities between $\{a_i, b_i, c_i\}$ and $\{a_j, b_j, c_j\}$.
- There is a homogeneous set of many P_3's with respect to these 2^9 possibilities.
- In each of the 2^9 cases, we find many disjoint P_3's, a clique, or a biclique.



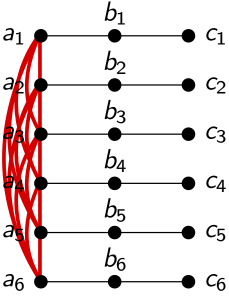
Finding subgraphs (hardness proof)

Lemma

Let $\mathcal{H}$ be a hereditary class of graphs that is not matching splittable. Then at least one of the following is true.

- $\mathcal{H}$ contains every clique.
- $\mathcal{H}$ contains every biclique.
- For every $n \geq 1$, $\mathcal{H}$ contains $n \cdot K_3$.
- For every $n \geq 1$, $\mathcal{H}$ contains $n \cdot P_3$.

- Consider many vertex-disjoint P_3's.
- For every $i < j$, there are 2^9 possibilities between $\{a_i, b_i, c_i\}$ and $\{a_j, b_j, c_j\}$.
- There is a homogeneous set of many P_3's with respect to these 2^9 possibilities.
- In each of the 2^9 cases, we find many disjoint P_3's, a clique, or a biclique.



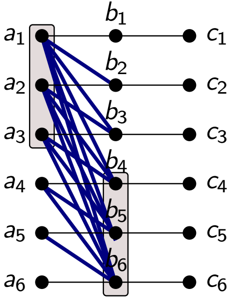
Finding subgraphs (hardness proof)

Lemma

Let $\mathcal{H}$ be a hereditary class of graphs that is not matching splittable. Then at least one of the following is true.

- $\mathcal{H}$ contains every clique.
- $\mathcal{H}$ contains every biclique.
- For every $n \geq 1$, $\mathcal{H}$ contains $n \cdot K_3$.
- For every $n \geq 1$, $\mathcal{H}$ contains $n \cdot P_3$.

- Consider many vertex-disjoint P_3's.
- For every $i < j$, there are 2^9 possibilities between $\{a_i, b_i, c_i\}$ and $\{a_j, b_j, c_j\}$.
- There is a homogeneous set of many P_3's with respect to these 2^9 possibilities.
- In each of the 2^9 cases, we find many disjoint P_3's, a clique, or a biclique.



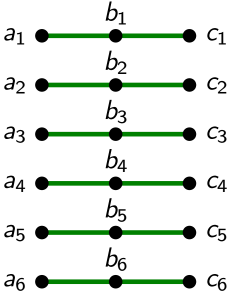
Finding subgraphs (hardness proof)

Lemma

Let $\mathcal{H}$ be a hereditary class of graphs that is not matching splittable. Then at least one of the following is true.

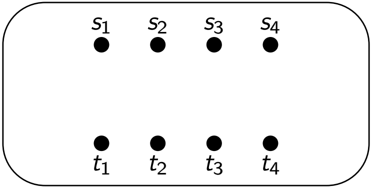
- $\mathcal{H}$ contains every clique.
- $\mathcal{H}$ contains every biclique.
- For every $n \geq 1$, $\mathcal{H}$ contains $n \cdot K_3$.
- For every $n \geq 1$, $\mathcal{H}$ contains $n \cdot P_3$.

- Consider many vertex-disjoint P_3's.
- For every $i < j$, there are 2^9 possibilities between $\{a_i, b_i, c_i\}$ and $\{a_j, b_j, c_j\}$.
- There is a homogeneous set of many P_3's with respect to these 2^9 possibilities.
- In each of the 2^9 cases, we find many disjoint P_3's, a clique, or a biclique.



Disjoint paths

k-Disjoint Paths
Input: graph G and pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$.
Task: find pairwise vertex-disjoint paths $P_1, \ldots, P_k$ such that P_i connects s_i and t_i.



NP-hard, but FPT parameterized by k:

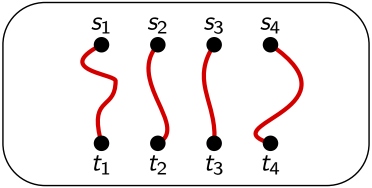
Theorem [Robertson and Seymour]

The k-Disjoint Paths problem can be solved in time $f(k)n^3$.

We consider now a maximization version of the problem.

Disjoint paths

k-DISJOINT PATHS
Input: graph G and pairs of vertices $(s_1, t_1), \dots, (s_k, t_k)$.
Task: find pairwise vertex-disjoint paths $P_1, \dots, P_k$ such that P_i connects s_i and t_i.



NP-hard, but FPT parameterized by k:

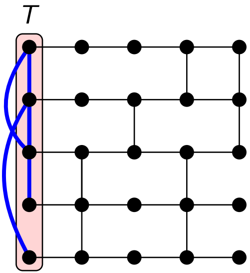
Theorem [Robertson and Seymour]

The k-DISJOINT PATHS problem can be solved in time $f(k)n^3$.

We consider now a maximization version of the problem.

Disjoint paths

MAXIMUM DISJOINT PATHS
Input: supply graph G, set $T \subseteq V(G)$ of terminals and a demand graph H on T.
Task: find k pairwise vertex-disjoint paths such that the two endpoints of each path are adjacent in H.

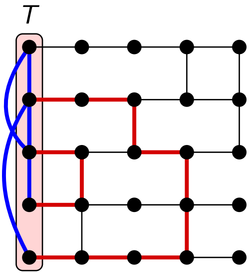


Can be solved in time $n^{O(k)}$, but W[1]-hard in general.

MAXIMUM DISJOINT $\mathcal{H}$-PATHS: special case when H restricted to be a member of $\mathcal{H}$.

Disjoint paths

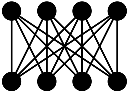
MAXIMUM DISJOINT PATHS
Input: supply graph G, set $T \subseteq V(G)$ of terminals and a demand graph H on T.
Task: find k pairwise vertex-disjoint paths such that the two endpoints of each path are adjacent in H.



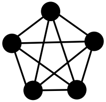
Can be solved in time $n^{O(k)}$, but W[1]-hard in general.

MAXIMUM DISJOINT $\mathcal{H}$-PATHS: special case when H restricted to be a member of $\mathcal{H}$.

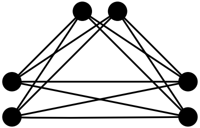
Maximum Disjoint $\mathcal{H}$-Paths



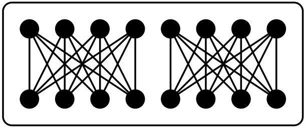
bicliques:
in P



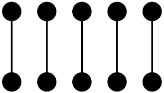
cliques:
in P



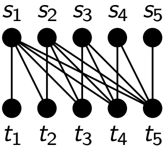
complete multipartite graphs:
in P



two disjoint bicliques:
FPT



matchings:
W[1]-hard



skew bicliques:
W[1]-hard

Maximum Disjoint $\mathcal{H}$-Paths

Questions:

- Algorithmic: FPT vs. W[1]-hard.
- Combinatorial (Erdős-Pósa): is there a function f such that there is either a set of k vertex-disjoint good paths of a set of $f(k)$ vertices covering every good path?

Maximum Disjoint $\mathcal{H}$-Paths

Questions:

- Algorithmic: FPT vs. W[1]-hard.
- Combinatorial (Erdős-Pósa): is there a function f such that there is either a set of k vertex-disjoint good paths of a set of $f(k)$ vertices covering every good path?

Theorem [M. and Wollan]

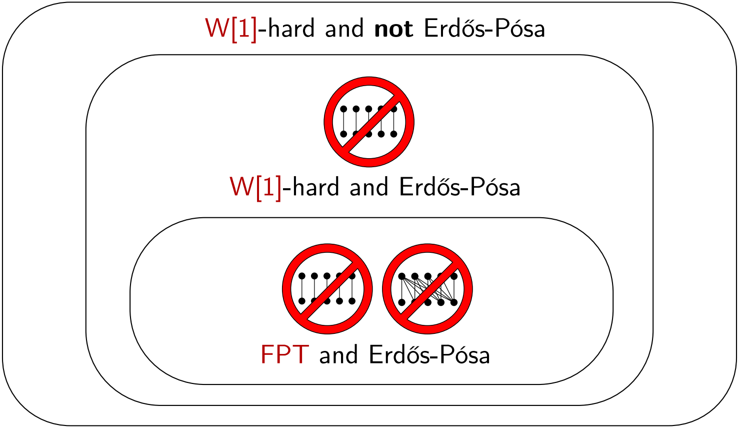
Let $\mathcal{H}$ be a hereditary class of graphs.

1. If $\mathcal{H}$ does not contain every matching and every skew biclique, then Maximum Disjoint $\mathcal{H}$-Paths is FPT and has the Erdős-Pósa Property.
2. If $\mathcal{H}$ does not contain every matching, but contains every skew biclique, then Maximum Disjoint $\mathcal{H}$-Paths is W[1]-hard, but has the Erdős-Pósa Property.
3. If $\mathcal{H}$ contains every matching, then Maximum Disjoint $\mathcal{H}$-Paths is W[1]-hard, and does not have the Erdős-Pósa Property.

Maximum Disjoint $\mathcal{H}$-Paths

Questions:

- Algorithmic: FPT vs. W[1]-hard.
- Combinatorial (Erdős-Pósa): is there a function f such that there is either a set of k vertex-disjoint good paths of a set of $f(k)$ vertices covering every good path?



Summary

Dichotomy results:

- P vs. NP-hard or FPT vs. W[1]-hard.
- For a fixed graph H or (hereditary) class $\mathcal{H}$.

Considered problems:

- H-FACTOR
- H-DECOMPOSITION
- H-COLORING
- H-PACKING
- $\textsc{Hom}(\mathcal{H}, -)$
- $\#\textsc{Hom}(\mathcal{H}, -)$
- $\#\textsc{Sub}(\mathcal{H})$
- $\textsc{Sub}(\mathcal{H})$

Conclusions

- For numerous problems, we can prove that every fixed graph (or graph class) is either easy or hard.
- Good research programs: easy to formulate, hard to solve, but not completely impossible.
- Possible outcomes:
  - Everything is hard, except some trivial cases.
  - Everything is hard, except the famous known nontrivial positive cases.
  - Some unexpected easy cases are found.
- Requires attacking the problem both from the algorithmic and the complexity side.